

COMP103P App Project

“TermSeeker”

Group Report

April 27th, 2016

Daniil Gannota
zcabgan@ucl.ac.uk

Berat Baran Cevik
berat.cevik.14@ucl.ac.uk

Client: Mark Shuttleworth (m.shuttleworth@ucl.ac.uk) from UCL School of European Languages, Culture and Society.

Developed by **Team 8** for UCL, Engineering Department.

TABLE OF CONTENTS

I.	Contents	2
II.	List of Figures	3
III.	List of Tables	3

CONTENT

1.	Introduction	4
1.1.	Team	4
1.2.	Vision	4
1.3.	First MoSCoW	5
2.	Team Management	6
2.1.	Step by Step Guide	6
2.2.	Distribution of Tasks	7
2.2.1.	Research	7
2.2.2.	Problems	7
2.2.3.	New Requirements	7
2.3.	Software	8
3.	Development	9
3.1.	Technical Requirements	9
3.1.1.	Use Cases	9
3.1.2.	User Scenarios	11
3.1.3.	Class Diagram	12
3.2.	Android Version	12
3.2.1.	Must Have	13
3.2.1.1.	Graphical Interface	13
3.2.1.2.	Progress	13
3.2.1.3.	Templates	14
3.2.1.4.	Search in Browser	14
3.2.1.5.	Limitations	14
3.2.1.6.	Debugging	14
3.2.2.	Should Have	14
3.2.2.1.	Limitations	15
3.2.2.2.	Debugging	15
3.2.3.	Could Have	16
3.2.3.1.	Debugging & Limitations	16
3.3.	iOS Version	16
3.3.1.	Learning	17
3.3.2.	Ending	17
4.	Google Search	18
4.1.	Search Engine	18
4.2.	Connection	18
4.2.1.	Private Keys.....	18
4.2.2.	Rendering	19
4.2.3.	Instructions	20
4.3.	Programming Methods	21

4.4. New Concept	21
5. Meetings	22
6. Appendix	24
6.1. Todoist	24
6.2. GitHub Repository	24
6.3. Slides for the Client	25
6.4. Logos	29
6.5. Excel File	30

LIST OF FIGURES

1. Figure 1: Development Cycle	4
2. Figure 2: MoSCoW (version 1)	5
3. Figure 3: Class Diagram	12
4. Figure 4: Interface Draft	13
5. Figure 5: Example of Private Keys	19
6. Figure 6: Example WebView Code	19
7. Figure 7: Example of Policy Breaker	20

***Note: figures in Appendix are unspecified.

LIST OF TABLES

1. Step by Step Guide	6
2. Use Case 1	9
3. Use Case 2	9
4. Use Case 3	10
5. Use Case 4	10
6. Use Case 5	10
7. Use Case 6	10
8. Meeting 1	22
9. Meeting 2	22
10. Meeting 3	22
11. Meeting 4	23
12. Meeting 5	23
13. Meeting 6	23

***Note: tables in Appendix are unspecified.

The report is fully written and edited by Daniil Gannota.

1. INTRODUCTION

1.1 Team

Overall, this project has helped us obtaining invaluable skills and developed further understanding of the computing world. We had been largely inexperienced programmers, and so spent significant time researching, understanding where things got wrong, changing the strategy and repeating everything again, *see Figure 1*.

Interesting, by doing just one project, we had to research so many areas. Although it might not be reflected in the final submission, the overall result, including googling and reading the most sophisticated methods, has been a personal success.

Daniil Gannota @ (tasks)

- documentation
- bi-weekly reports & the final report
- meetings & negotiation
- team management
- iOS development
- design amends
- Custom Search Engine implementation

Berat Cevik @ (tasks)

- documentation
- meetings
- first version Android development
- design amends
- video

To see more, go to Section 6.1.

1.2 Vision

The main purpose of the application is to provide a modified way to search for terminology. It adds refinements in language codes as well as gives a list of keywords that can be used to improve the final results.

Our team had a confused understanding of the project for the first time. This is due to similarities with the regular search. However, various meetings with our client and a person from IT-department helped clarifying the subtleties. Therefore, requirements, for instance MoSCoW, have been constantly amending throughout the project. If we discovered a limitation or a computing difficulty, we immediately contacted the client, negotiated new terms and reflected this in the documentation.

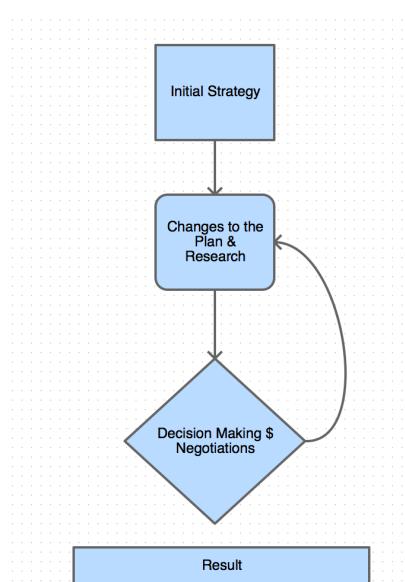


Figure 1: Development Cycle

1.3 First MoSCoW

This is the first version of MoSCoW list that has been confirmed by our client, including all the initial requirements:

- ✓ stands for completed tasks
- ❖ stands for partially obtained
- not finished

MoSCoW

Must Have:

- ✓ Search for Terminology – Google (or any other) search engine implemented into mobile application;
- ✓ Up to 60 Languages – allow user to choose language, consider words of the same spelling but different meaning;
- ✓ Android/iOS Application
- ✓ Advanced Search – example of Google's advanced search;

Should Have:

- ✓ Star Ranking – allow user to star rank sources from 1 to 5;
- ❖ Filtering – relevance by default, other options: date, meta-tagging, star-ranking;
- ✓ User History – save locally history of searches and rankings;

Could Have:

- Different Search Engines – allow user an option to choose between search engines;
- ✓ Share Option – including emailing and other preferred social networks;
- Smart Results Appearing – preferable source appears first depending on star ranking, unranked sources appear randomly so that the user can score them;
- ✓ Meta-Tagging – *to be further discussed*;
- ✓ Sponsored Links – must be filtered out from the search, no advertisement;
- ✓ Open Browser Window – allow an option to open search in a browser window, *to be further discussed*;

Won't Have:

- ✓ User Registration – no log in / sign in, user history is saved locally on device, *star ranking to be further discussed with IT client*;
- ✓ In-App Purchases

Some of the ideas have been deleted/edited accordingly to the client choice. The MoSCoW presented is valid for this term only as the next will require further discussions.

Figure 2: MoSCoW (version 1)

NOTE: The point stating “Different Search Engine” can be counted as to be partially completed. There is no such option available in our application, as we were working with Google API only (see in Section 4 of the report), however, the client can always add additional websites to the search we implemented. This is done on the server side.

2. TEAM MANAGEMENT

2.1 Step by Step Guide

This is a guide we had confirmed with the client and had been following at the very start of the project:

Step by Step Guide
The report structure as follows: major implementation as a heading, following small iterations as steps numbered in order.
Search for words:
<ol style="list-style-type: none"> 1. Use Java and either Objective C or Swift languages for platforms Android and iOS, respectively. 2. Implement Google into mobile application. 3. Add possibility to choose one language whilst having “English” set by default. This option affects google search that will be done, e.g. choosing English will mean search through google.co.uk, choosing Russian – google.ru. 4. Add table with the key words in .xml format (using parsing of data). This will add the key words you are providing with the diagram, which is an Excel file. 5. Add attributes. This will mean to use ('inurl: ...' and such as this) google attributes in the search, ticked all on by default. Later implementation of Advanced mode will allow to tick off/on those attributes. 6. Add possibility to choose second language. There are a few questions to be asked about this particular option. Firstly, Star-Ranking requires database in order to have average score ordering of resources. At the same time, we will need to research the way Google results can be affected by our Star-Ranking system. Secondly, later iteration of Star-Ranking system will mean that users may view the application in a different way: for example, affecting the most popular websites that the application WILL display and YOU WANT it to display. Basically, the application may evolve in something completely different, though this is just my opinion. However, this is not a part of “Must Have” step for now, so is later to be determined. 7. Deploy for IOS and Android – and test it. Might take some time, however, to eliminate some bugs and make search more stable.
Advanced Search mode:
<ol style="list-style-type: none"> 1. Add possibility to choose attributes. As a part of our work, we are to research attributes that Google is having, choose most preferable in our opinion and confirm them with you. For instance, when a user inputs word ‘medicine’ – where is the search performed? Inside the url (domain) of website? Or inside their texts? That is what is referred as ‘google attributes’.

2.2 Distribution of Tasks

Section 1.1 shows a brief overview of the work each member in our team have been doing. More detailed explanation is added at Section 2.3.

2.2.1 Research

At the very start of the project, our team was researching the requirements from the original hand-out. The whole app-development was some we had never done before, so it would be fair saying that the research was being done incorrectly. As from today, we now know where to start looking for information and services, we learnt a lot of methods and limitations they impose. It is also fair to say that we could be a lot more productive and have outstanding results if not the inefficient work that was occurring.

The first problem we were faced with was whether we had to program in Cordova language or choose something completely different. Thus, the first was all about learning that language while researching alternatives. As a result, our team had though it would be better programming for Android and iOS platform separately.

Switching to the new IDEs and languages, our team spent time obtaining the necessary skill to start programming. It took us shorter time to learn Android Studio, but developing for iOS made us learning Swift and Xcode. The second platform had taken quite a time to be able to commence the work; considering simultaneous Android development, report and documentation, constant requirements negotiations, changes in the overall application looking and feeling, presentations held for the client and various limitations that we were discovering.

As a result, the development for iOS was stopped at a certain moment when a new way of computing the application was found, see Section 4.

Lastly, constant researches about server databases, which we were eventually advised not to include, and Google Services were quite ineffective as our team had not known what websites to use, thus simply googling and finding irrelevant information.

2.2.2 Problems

At the end, we have a list of problems accumulated:

- 1) External database (would solve many issues)
 - a) No filters
 - b) No proper meta-tagging
 - c) No top results
 - d) No smart appearing
- 2) Google API (which we did not use in the best way)
 - a) No analytical data observed
 - b) No private connection

2.2.3 New Requirements

Alongside with constant switching between different approaches, our team had to change documentation accordingly. This involved various emails and meeting with the client, negotiations and a new research.

When we agreed over a modified requirement, our team immediately edited documentation accordingly. This process repeated a few time, making the starting point too long to take. Additionally, we could not avoid documentation as it would prove to be compulsory for proper project handling.

2.3 Software

Throughout the project we have been using different tools to improve team communication and overall work experience.

To start with, we added all the MoSCoW requirement to an application service called Todoist. It was used to keep tract of overall development as well as raise concerns over new performance issues or limitations. Appendix 6.1 will have some of the screenshots from the server.

Furthermore, we used GitHub in order to work simultaneously on the same project. Appendix 6.3 shows some screenshots from the repository. It is worth saying that git does not work properly with large projects such as Android development. Therefore, it was almost always the case for manual application update.

3. DEVELOPMENT

3.1 Technical Requirements

As outlined earlier above, before starting this app project, neither of us had any experience in mobile app-development, and we decide to program for two platforms separately due to conveniences each IDE provides.

After we had confirmed the general procedure of app-development with the client, we started developing technical documentation that was meant to organise the structure of our application as well as enabling team work.

NOTE: Some of this was added to the bi-weekly reports, however, current report illustrates the most recent version. Moreover, these requirements were mostly developed for 'TermSeeker with External Browser' version, not the 'TermSeeker with CSE'.

3.1.1 Use cases

Use Case: Basic Search
ID: 1
Brief Description: The user can do basic search.
Primary Actor: User
Secondary Actor: None
Preconditions: Internet Connection
Main Flow:
<ol style="list-style-type: none"> 1. The user selects one or two language(s) for the search. 2. The user types the words in the search tab. 3. The user selects a keyword group. 4. The user makes basic search. 5. If the user chose one language, the results in the specified language would be shown. 6. If the user chose two languages, the results which have a content in both of the specified languages would be shown.
Post Conditions: None
Alternative Flow: If the user does not specifies a language, it will be set to English by default.

Use Case: Advanced Search
ID: 2
Brief Description: The user can do advanced search by specifying some words.
Primary Actor: User
Secondary Actor: None
Preconditions: Internet Connection
Main Flow:
<ol style="list-style-type: none"> 1. The user selects one or two language(s) for the search. 2. The user taps on the advanced search. 3. The user types the words in the advanced search tabs such as "all these words", "this exact word or phrase", "any of these words" and "none of these words". 4. The user makes advanced search. 5. If the user chose one language, the results in the specified language would be shown. 6. If the user chose two languages, the results which have a content in both of the specified languages would be shown.
Post Conditions: None
Alternative Flow: If the user does not specifies a language it will be set to English and by default.

Use Case: Language Selection
ID: 3
Brief Description: The user can choose one or two languages among language selections.
Primary Actor: User
Secondary Actor: None
Preconditions: None
Main Flow:
1. The user selects one or two language(s) for the search.
2. The user makes a basic or advanced search.
Post Conditions: Keep the language selection(s) until the user closes the app.
Alternative Flow: If the user does not specifies a language, it will be set to English by default.

Use Case: My Rating
ID: 4
Brief Description: The user can change the rating of the results or remove them.
Primary Actor: User
Secondary Actor: None
Preconditions: Internet Connection
Main Flow:
1. The user makes a basic or advanced search.
2. The user goes to the website that he/she selects.
3. When the user is done, he/she can rate the search from 1 to 5 on user history screen.
4. The user can change a rating or remove it from my ratings screen.
5. The user can rate a search or he/she can choose not to give a rating.
6. The user goes back to home page.
Post Conditions: None
Alternative Flow: The user can delete the ratings by tapping on "Clear Ratings".

Use Case: User History
ID: 5
Brief Description: The user can see his/her search history and can delete it.
Primary Actor: User
Secondary Actor: None
Preconditions: None
Main Flow:
1. The user taps on "User History".
2. The user selects a search that he/she has made.
3. The user can also rate this search.
4. The user goes to the selected website.
5. When the user is done, he/she goes back to user history page.
6. The user goes back to home page.
Post Conditions: None
Alternative Flow: The user can delete the user history by tapping on "Clear History".

Use Case: Settings
ID: 6
Brief Description: The user can set his/her default language.
Primary Actor: User
Secondary Actor: None

Preconditions: None
Main Flow:
<ol style="list-style-type: none"> 1. The user taps on “Settings”. 2. The user selects default language. 3. The user goes back to home page.
Post Conditions: The user’s preferences will be saved.
Alternative Flow: None

3.1.2 User Scenarios

Basic Search Scenario

1. Users opens the app.
2. User selects Language 1 and Language 2 as languages.
3. User types input in the search field.
4. User selects one of three categories of keyword groups.
5. User performs basic search.
6. The results are shown in a separate browser.
7. User taps one of the links.
8. When the user is done with the website, return back to the application.
9. User goes to user history.
10. User is able to rate a result.
11. User goes to my ratings.
12. User is able to send ratings to all possible social networks.
13. The user goes back to home page.

Advanced Search Scenario

1. User opens the app.
2. User taps on “Advanced Search”.
3. User selects Language 1 and Language 2 as languages.
4. User types input in the “all these words” tab.
5. User types input in the “any of these words” tab.
6. User types input and in the “none of these words” tab.
7. User performs advanced search.
8. The results are shown in a separate browser.
9. User taps one of the links.
10. When the user is done with the website, return back to the application.
11. User goes to user history.
10. User is able to rate a result.
11. User goes to my ratings.
12. User is able to send ratings to all possible social networks.
13. The user goes back to home page.

User History, Ratings and Settings Scenario

1. User opens the app.
2. User goes to “Settings”.
3. User sets default Language 1.
4. User goes back to home page.
5. User goes to “User History”.
6. User taps on the most recent website that were visited.

7. When the user is done with the website, return back to the application.
8. User deletes selected results by long tapping.
9. User deletes all the results by clicking a separate ‘Delete’ button.
10. The user goes back to home page
11. User goes taps and rates search results.
5. User goes to “Ratings”.
6. User taps on the most recently rated result.
7. When the user is done with the website, return back to the application.
8. User deletes selected results by long tapping.
9. User deletes all the results by clicking a separate ‘Delete’ button.
10. The user goes back to home page.

3.1.3 Class Diagram

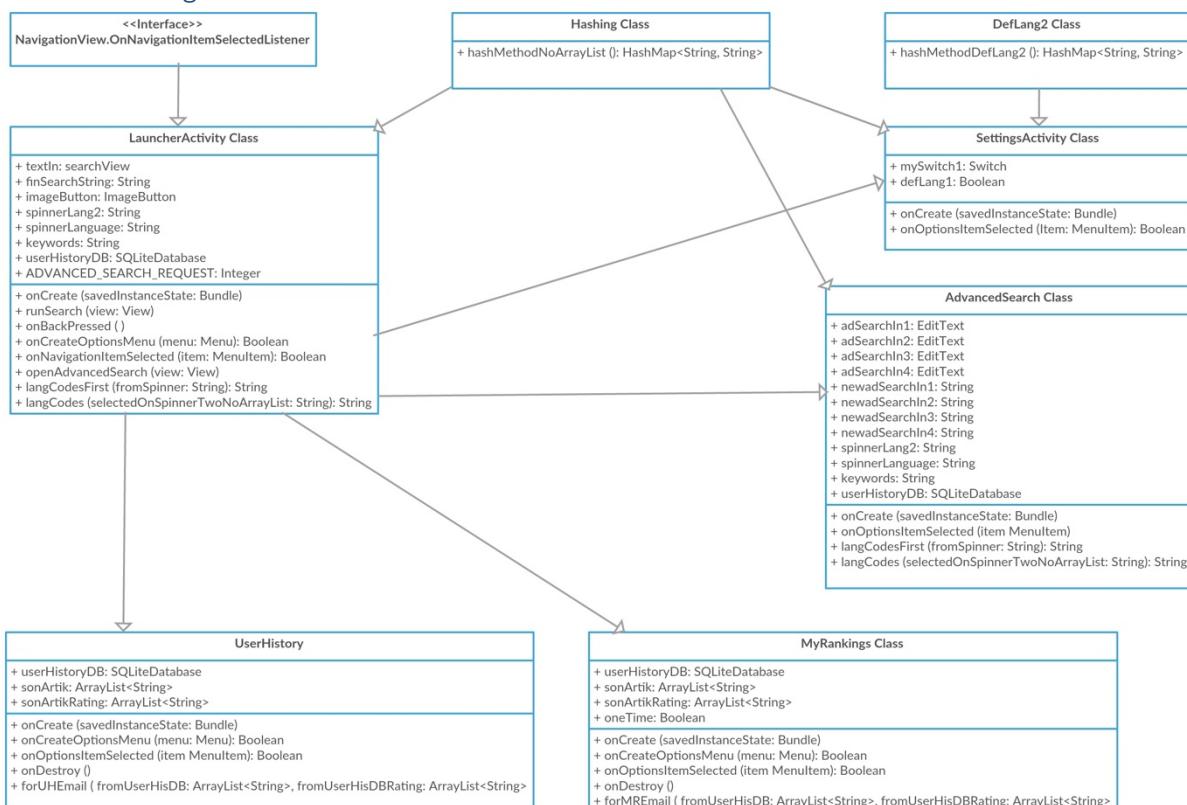


Figure 3: Class Diagram

3.2 Android Version

After confirming the “Step by Step” guide and UI draft with the client as well as writing the necessary documentation, we conducted a research about Android Studio. This helped the whole team to switch to Android development at any stage if urgently needed so. Our primal resource was a tutorial from Lynda.com. As new versions of Android Studio were released, for instance there was a large update from 1.5 to 2.0, we spent time checking functionality they had brought to ensure safe and stable version of the final application.

NOTE: as a matter of fact, not all updates for Android Studio smoothly adapted to our code. Some of them required for a re-build of the Gradle file, others just a few little changes to the

functions. However, one of the recent versions, 2.0 and then 2.1, had some issues with WebView functionality which caused unexpected crashes. Sequentially, our team had to rewrite certain aspects as to ensure the application is stable even while the bugs are not fixed by Google itself.

3.2.1 Must Have

As a matter of priority, we decided to start the development with “Must Have” requirements from our “MoSCoW” list. Additionally, throughout the development process we largely benefited from other documentation as well as the managerial tools described earlier as all of it allowed cohesive team work.

3.2.1.1 Graphical Interface



Figure 4: Interface Draft

As we had confirmed the graphical interface with our client, the first programming step was to use the templates that Android Studio includes to build standard activity pages. This is the original design pattern that was confirmed, see *Figure 3*.

Afterwards, we changed the colour of the theme to blue tones, regarding the meeting we had held with our client and his team. Additionally, we added a navigation drawer to enhance the user experience with menu items “User History”, “My Ratings” and “Settings”. We also added keywords for both languages on the main screen, since keywords are a distinctive feature of this application.

3.2.1.2 Progress

In terms of other requirements, we needed to implement basic Google Search, Google Advanced Search both with 64 languages option. These 64 languages were provided by our client in an Excel file. We spent quite a long time on reading Excel files through the application by parsing the data to it; however, as our team could not resolve this issue, we decided to copy and paste these languages in an internal xml file. The downside of this is that it occupies a little more storage than the usual Excel file, see Section 6.5.

Another difficulty we had was about displaying the languages, i.e. linking two spinner items so that the languages in the second spinner depended on the language selection of the primal. For instance, if user selects Chinese in the first spinner item, then all the language names in the secondary spinner will be displayed in Chinese.

In order to achieve that, after some research on Java, we decided that the best way to implement this was to use a hash map. This helped us to manage the language codes and the language names easily. We defined this hash map in a separate class. Additionally, we defined certain functions to manage this hash map using them for keywords section as well.

The search results were supposed to be shown in a WebView inside the app, however, we could not implement this at the beginning. Therefore, we used an external browser support instead to implement basic and advanced searches.

For Google Advanced Search, we created a separate activity. In order to implement Google Advanced Search effectively, we performed certain experiments with Google Advanced search to understand the principle behind it. Afterwards, we added this functionality inside the advanced search activity class.

Later, we managed to create a Google Custom Search Engine through which users are able to have results of the input inside the application as our client requested.

3.2.2.3 Templates

When creating the graphical user interface of the app, we used interface items provided by Android Studio. Apart from that, we did not use any prepared templates or design patterns. Additionally, we created our own design considering clients' requirements as well as confirming it.

3.2.2.4 Browser Search

In order to implement input searching in a browser, we researched how Google uses URL to affect the search results; and by manipulating the URL that we pass to the external browser, we allowed users to make both basic and advanced search through the application. Moreover, our team benefited from this URL manipulation later, when we started using a web view to display the search results.

3.2.2.5 Limitations

The major limitation was that we did not add support of inside-application results search. Therefore, we could not actually add preferable filters or refinements that had been originally required. This was discussed with the client, changing corresponding work planning accordingly.

3.2.2.6 Debugging

This process was including debugging by the whole team, as well as searching for unresolved bugs. For example, while testing we noticed that if the user enters more than just one word in one of the advanced search bars, they would appear in Google Search as one word. The overall procedure took us about 3 days.

3.2.3 Should Have

As soon as we finished the "Must Have" requirements, we moved to "Should Have" which includes: star ranking system, filtering and user history. Filtering, however, was not added to 'TermSeeker with External Browser' version of the application. This is a problem followed from limitations of "Must Have" work.

For user history we decided to use Android SQLite database in order to store user's search history and star ratings. Therefore, this required our team to research and learn the bases of database prior to actual implementation. When the user makes a basic or an advanced

search, the words and the keywords that they have searched for are sent to SQLite database. However, in the recent version we only stored input without the keywords. This is due to the fact that originally the idea was to give an option to choose from a large list of keywords. Afterwards, our client stated that he would like to have these keywords to be separated into three categories. Therefore, our original database started saving a long list of works along with the input. The best solution to this issue was to either completely change the database, or name the categories. However, as our client had not provided any instructions accordingly this problem, we decided to eliminate keywords from “User History” savings for the time being.

Users are able to delete one search input by long tapping on the item. A confirmation dialog will appear. Additionally, users are able to search an item again by tapping on it inside the “User History” page. Lastly, users are able to delete all of the searches by on a separate button at the top right on the same page.

As a distinctive feature of application as well as another “Should Have” requirement, we implemented star rating system which allowed users to rate their search results and see them on “Rating” screen. We added one more button to rate each item. The available ratings are from 1 to 5 stars. If user gives 0 rating to a search result, the application will assume that the user wills to delete the rating. Therefore 0 score corresponds deleting an item from star ranking page.

Furthermore, we used SQLite Database to store the ratings too. Every search input is stored with a unique score, which is 0 by default. And as users change the score, the inputs will be appearing/disappearing on the corresponding page.

3.2.3.1 Limitations

As we stated earlier above, filtering option was not added to the browser searching. Moreover, we were supposed to collect star ratings from users after they have visited a URL. However, it was not possible for us to implement such feature as due to the same reasons concerning filters. Although we made a quite comprehensive research about that point, we were unable to implement it due to its complexity and personal inexperience. However, we manage to implement WebView as a result of this research.

Furthermore, we were supposed to use a remote database to store star rankings so that once one user ranked a source, all of the other could see the average rating. This would be managed as a database on an external server. Our team was advised not to use this feature as due to the amount of work it required. Additionally, we contacted UCL department and talked to other advisors, and all told to negotiate a different functionality. Thus, our team acted accordingly, amending the requirements.

3.2.3.2 Debugging

Debugging phase for “Should Have” requirements was very challenging. We spent a lot of time on Google’s Developer website to find the best features and solutions to implement in our app. This was also considering server databases research.

Furthermore, we have two versions of our application to correspond to different tasks. This point will be further evaluated in Section 4. The reason we did not merge the two solutions into one application is that they use quite different techniques, and therefore had required major changes for complementary activities, such as user history. Hence, this would in some way overwrite the whole code patterns. Our team specifically asks to check both versions for better assessment.

3.2.4 Could Have

Since we had finished developing most of “Must Have” and “Should Have” requirements, for the rest of the time we tried to implement as many “Could Have” requirements as possible. You can check the whole list of completed and not tasks in the MoSCoW section on page 5.

Thus, we implemented email option which allows users to send their user history and ratings via any social utility. When users tap on the email icon at the top of user history and my ratings pages, the application asks them to choose the preferable option from various available. When the user chooses one of options, the data is passed along with the ratings in parentheses. This feature improves apps interaction with humans.

As an additional function we added settings activity to let the users to set a primary default language, so that this will appear the same every time the application launches. We also tried to add this feature to select a default language for the secondary option, but when we added it our client explicitly asked to remove it. This was due to the fact that secondary language field must start with “no language selected” option. However, we spent a lot of time configuring the second spinner, as it is linked to the first; i.e. if the default for Language 1 was changed, Language 2 was automatically translated into the new language.

As a matter of fact, we were initially using default settings activity class provided by Android Studio. However, we quickly decided to create our own activity class in order to optimise the code by getting rid of the trivial methods Android Studio’s settings activity is using. As ‘shared preferences’ is the most preferable way of implementing settings, we needed to conduct some research about it. Afterwards, we succeeded in our final result.

3.2.4.1 Debugging & Limitations

Skipping a separate “Limitations” section due to “Could Have” list of queries remaining largely incomplete, we tested the final application on different Android devices as well as used various emulators available. We discovered a performance issue: on the devices with lower specifications the application takes a few seconds on the very first launch to create the necessary database. The solution would be to use background activity, which was not implemented due to time constraints.

3.3 iOS Version

This version of our application has not been submitted as we abundant the development at a stage when it was too unstable having only basic features while Android version required extra support due to its own limitations’ problems.

3.3.1 Learning

Our team spent time learning Swift language. Now we know that Xcode supports C too, but not back then. Therefore, an amount of time was certainly wasted if thinking about final result, however, provided invaluable experience. We now know Swift!

Our next step after learning Swift was to learn Apple's IDE – Xcode. This took a little while as it differs in some subtle ways from what Android Studio allows.

Overall, learning period took longer than originally anticipated as due to urgent problems concerning Android development. Constant switching as well as change of requirements had induced unstable application.

3.3.2 Ending

At the time when we had to decide whether to concentrate on Android platform or try to revive iOS, but this would have us presenting two applications with main features only.

This application was unstable due to three reasons:

1. Google Search
2. Database
3. Parsing Data

At that time, our team was negotiating different terms with the client about requirements. However, iOS application was originally structured in a way so that it implements the actual Google Search and renders the results in a specified format. However, we only discovered Google API services at the very end, leaving to a choice of implementing into either iOS or Android.

The other two factors, Database and Parsing, are closely linked as database was supposed to store some information parsed from Excel file, see Section 6.5. This was a challenging part, and the final result was working. The reason they could not exist, however is again linked to the Google API services. More about Google Services is in Section 4.

4. GOOGLE SEARCH

4.1 Search Engine

In order to access search results inside our application, we had to link Google Services with it. Therefore, our team used application programming interfaces (APIs) that Google provides for developers. This enables communication between any Google Service, such as Custom Search Engine which is using Google Search, in our case, and displays the necessary data in the specified format.

However, we have to admit that this implementation was the most challenging part of the project. Not only this took about a week, overall, to research, but is generally used on websites only. Hence, most of the tutorials explained how to add a JavaScript file to your webpage, and did not provide information about app-development. Thus, we tried to convert/link JavaScript code into/with our Java application, but this did not work properly. Likely, eventually, we found a way to add a direct connection to the application.

Generally, the way it works is that there are certain steps to be followed: creating a search engine, configuring it, enabling connection, modifying Android application and linking the processes. First of all, GoogleCloud Platform provides Google APIs that can be created and configured. Secondly, we had to research different methods that are used to establish safe and protected communication between any device and one particular search engine. Lastly, we amended our application so that it connects to the service only when the user asked so, modifying user interface accordingly.

4.2 Connection

There are a few ways in which Custom Search is connected to Android or any other application. The simplest solution was to create a public server and open it inside the application using WebView library. Linking the Android application with the server required enabling of public server ID. This ID allows only users who know it to access the service. Therefore, we added it to the code, by passing 'cx' to URL of the engine, so that the connection occurs automatically.

Nevertheless, we have also tried to enable more sophisticated connection which would both protect our customers and display the data in the best mobile format.

4.2.1 Private Keys

The other way to establish communication between a device and a service would be to use a private key. The idea is that only users who obtained the key are able to access the servers. By comparison, public link can be used by anyone as long as the address is known whilst this method requires constant/at least one-time authorisation. To do so, we created '.keystore' file as well as enabled the necessary login procedure on the Custom Search side. Overall, as our application had the keystore file signed to it, only this program could be using the Custom Search Engine.

API keys			
	Name	Creation date	Type
<input type="checkbox"/>	Android key (auto created by Google Service)	Apr 27, 2016	Android
<input type="checkbox"/>	Android Key -- Default	Apr 20, 2016	Android
OAuth 2.0 client IDs			
	Name	Creation date	Type
<input type="checkbox"/>	Android Client -- Admin Danill Gannota	Apr 20, 2016	Android
			842327975997-t7apeoctmtvcik26ctveg9vrmu9qdrnf.apps.googleusercontent.com

Figure 5: Example of Private Keys

We have been researching these procedures provided by Google, for instance, OAuth 2.0 authorisation. Additionally, there is another type of private connection – API keys. The main difference lies within conveniences of either checking the login details at the launch of application, or every time while loading response to new input.

Unfortunately, our team could not achieve decent private connection. One of the reasons is that we are completely new to this service, therefore all step performed are taken from programming blogs which can be inconsistent in explanations. Moreover, Google has just released quite an unstable update to Android Studio which slightly breaks WebView library. This is one of the reasons we could not test all solutions provided on the internet – any significant level of complication to WebView makes applications crash. As a result, private connection remained unresolved.

4.2.2 Rendering

```
public class RestauranteUniversitarioActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        WebView cardapio = (WebView)findViewById(R.id.web_engine);
        cardapio.getSettings().setJavaScriptEnabled(true);

        String data = "";
        Document doc = null;
        try {
            doc = Jsoup.connect("http://www.iguatu.ce.gov.br/").get();
        }
        catch (IOException e) {
            e.printStackTrace();
        }

        Elements content = doc.getElementsByClass("dest-left");
        if(content.size() > 0) {
            data = content.get(1).text();
        }
        cardapio.loadData(data, "text/html", "UTF-8");
    }
}
```

Figure 6: Example WebView Code

As a way to optimise web-version of the Custom Search Engine, we have been digging all over the internet to find a method which analyses pages and only renders what is explicitly specified.

The easiest solution was to use WebView and then '@Override' it with a specified 'div' section. Ideally, it only shows the part of webpage that is within the 'div'. However, in our case we had no 'ids', and instead had to specify 'class' of the 'div'. The key different is that 'ids' are unique while 'classes' can be of similar name which can render additional files. Moreover, `webView.getElementsByClass('class')` function receives an array of elements

which must then be outputted singularly using a loop. It is worth noting that a lot of our research was based on this part, and unfortunately the discovery of this little limitation required long hours of testing and googling.

Eventually, we came with a better solution in which we are using Jsoup to parse only certain code to an array of elements, *see Figure*. The array is then converted into the necessary html format and loaded to the user. We specified all the necessary “try catch” exception throws for better debugging process. As a result, this did not work, supposedly, due to protection each web-page includes.

As our knowledge is limited, we have tried most, if not all, of the examples from the internet. Moreover, this rendering only works with AsyncTask, which enables background activity in the application, or IntentService; our team tried both of ways. The reason why Android requires background activity is due to loading time – it will crash the application if network loading may take too long potentially. Therefore, we even tried to modify Google’s policy while testing the performance issues:

```
StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder().permitAll().b  
StrictMode.setThreadPolicy(policy);
```

Figure 7: Example of Policy Breaker

This coding part is rather to be avoided as it breaks all the protective limitations instructed by Google. However, we even tested this example to ensure that everything was used.

As a matter of fact, the last example helped us to go the furthest in the rendering process, although we were faced with the last and main issues – web-page protection. Most of the pages online do not allow to access just a specified part of the page. The error was either “Cannot call determinedVisibility()” or “java.io.FileNotFoundException”. Bloggers suggested that the problem lies within private connection, which we could not implement as described in Section 4.2.1.

4.2.3 Instructions

In order to do the implementation without disturbing our client, we decided to register the search engine on one of our personal Google accounts. This means that the application will need to be re-connected to our client’s account to have full control over the Custom Search Engine. Our team will have a voluntary meeting after UCL’s deadline for the project, thus, the customer will be provided with the information about registering and linking services. Additionally, we will provide some advices about the platform and explain ways the output results can be formatted and improved, as well as the ways statistical data is collected about the users.

Furthermore, we will recommend Google Analysis Tool that records statistics about input, searches and rates – potentially, anything that is in the application. Therefore, our client will have an idea about how to use features we have implemented, for instance User Rating system, and improve application accordingly.

Last, there is a limited number of searches that can be done per day – 100. Any additional 1000 will cost around 5 dollars. Additionally, there is an option for paid version of Google Services which we have not tested and are unable to comment on.

4.3 Programming Methods

Google's CSE service provides most of the utilities our client was asking for. However, there is one nuance – it's static and applies to all users. In our free version we could only specify 'keywords' or 'refinements' that once added will be displayed to all users, and therefore cannot be removed by them.

The purpose of the application is to provide customers with the choice of languages and keywords. They can have either one language selected, or both; it also allows to choose out of three categories of keywords. Thus, when the user selects primal language, it will add "site=.com" or "site=.ru" depending on the country – this is a parameter that filters only the links ending with specified country value. Choosing the secondary language will add a second "site=.smth" to the input, with 'OR' parameter allowing to have both countries in the search.

Keywords in our application work differently from what Google has in the CSE setup profile. As they only have static keywords, we found a way to add 'refinements' manually. This is done by passing the chosen keywords inside our application and allowing them to influence the top results. The both methods are performed by passing additional attributes to final url. For example, "gsc.q=" affects the input, while "gsc.ref=more" prioritizes the top results.

4.4 New Concept

In order to accommodate the new features coming from CSE, we had to largely modify the concept of the application. That is to say, in our submission we have provided you with two versions: one is using CSE, another just the browser.

Ideally, our team would like to combine these versions, allowing the user to pick ways in which results are displayed. However, they were not merged due to time constraints. We could easily add 'Browser' option inside of our CSE application, but our client has not confirmed this design solution. Therefore, we will send both file and ask whether one is enough, or they have to be single.

It is also meant to show different approaches. As we were not experienced enough to find the best solution at the very beginning, despite thorough documentations and various meetings, this resulted into some additional changes. Nevertheless, it is still a great experience for our team.

5. Meetings

Meeting Number: 1
Met with: Mark Shuttleworth (Client)
Meeting Duration: 1 hour approximately
Summary of the Meeting: Initiations. Discussion about the app. The meeting with the client gave an overview of the project. We also conducted the development path that our team will be following.

Meeting Number: 2
Met with: Pablo (IT co-worker) from Mark Shuttleworth's Team
Meeting Duration: 45 minutes
Summary of the Meeting: Initiations. Pablo clarified general application purpose as well as specified programming challenges, for example, we noticed that there are two ways in which a user can affect Google search: by choosing right key words, given by the client, and (or) by selecting advanced search attributes.

Meeting Number: 3
Met with: Mark Shuttleworth
Meeting Duration: 20 minutes
Summary of the Meeting: Discussion and evaluation of the meeting with Pablo. Discussions on the requirements considering the researches we had made. Clarification of "MoSCoW".

Meeting Number: 4

Met with: Mark Shuttleworth

Meeting Duration: 20 minutes

Summary of the Meeting: Clarification of certain points about app-development as well as design decisions. A few new features have been discovered, previously undescribed in “MoSCoW”.

Meeting Number: 5

Met with: Mark Shuttleworth and His Team

Meeting Duration: 1 hour 10 minutes

Summary of the Meeting: Initiations. Presentation about the current situation of the app, development process and development tools.

Meeting Number: 6

Met with: Mark Shuttleworth

Meeting Duration: 30 minutes

Summary of the Meeting: Initiations. Presentation about the current situation of the app, development process and development tools.

***Note: there was one more meeting which was a presentation of our current work for the students. It was a query from our client. See Section 6.3.

6. Appendix

6.1 Todoist

Just a few screenshots of our team work from Todoist.

- | | |
|---|--|
| ✓ Link languages to all data | ✓ Report-2 |
| ✓ Add Google attributes | ✓ Report-1 |
| ✓ Add .xml file | ✓ Write the bi-weekly report |
| ✓ Allow to choose keywords | ✓ Design concepts |
| ✓ Implement Google search [comment] | ✓ Create Presentation Slides |
| ✓ Enable Search Results inside the App | ✓ Write first prototype in Java |
| ✓ Android | ✓ Prototype User Interface |
| ✓ Final classes diagram [comment] 4 | ✓ Draw design concepts (a few to choose?) |
| ✓ Report-3 | ✓ Confirm UI with the client [comment] |
| ✓ User Scenarios | ✓ Draw UI sketch |
| ✓ Use Cases | |
| ✓ Pre-requirements | |
| ✓ Corrections in Turkish Dictionary | |
| ✓ Research Google Search | |
| ✓ A brief description [comment] 2 | |
| ✓ Draw sequential diagram and confirm it | |
| ✓ Arrange a meeting with IT guy [comment] 4 | |

6.2 GitHub Repository

 Testing Uploaded Code 1 month ago by danikgan 5 ▶	 Android Version 0.8 8 days ago by beratbarancevik 10 ▶
 Optimisation Issues 1 month ago by danikgan 25 ▶	 Android Version 0.7 9 days ago by beratbarancevik 4 ▶
 Android Version 0.2 1 month ago by beratbarancevik	 Android Version 0.6 9 days ago by beratbarancevik 56 ▶
 Unzipped properly 1 month ago by beratbarancevik 67 ▶	 Android Version 0.5 1 month ago by beratbarancevik 5 ▶
 Unzipped 1 month ago by danikgan	 Android Version 0.4 1 month ago by beratbarancevik 23 ▶
 Zip 1 month ago by beratbarancevik	 Android Version 0.3 1 month ago by beratbarancevik 32 ▶
 Android&iOS 1 month ago by danikgan	 Android Version 0.2 1 month ago by beratbarancevik 70 ▶
	 VM optimisation 1 month ago by danikgan 3 ▶

6.3 Presentation Slides

There was one more meeting which was a presentation of our current work for the students. It was a query from our client, we did voluntary work.

TermSeeker

by Daniil Gannota and Berat Cevik, Team 8

Introduction

- Google Search
- Key Words: three sets or singular selection
- Attributes: Advanced Search and Hidden Tricks
- Other: Ranking System, Filters, Statistical Data etc.

Requirements: MoSCoW

> MoSCoW

Must Have:

- ✓ Search for Terminology – Google (or any other) search engine integrated into mobile application;
- ✓ Up to 90 Languages – allow user to choose language, consider words of the same spelling but different meaning;
- ✓ Advanced Applets;
- ✓ Advanced Search – example of Google's advanced search;

Should Have:

- Star Ranking – allow user to star rate sources from 1 to 5;
- Filtering – relevance by default, other options: date, meta-tagging, star-ranking;
- User History – save locally history of searches and rankings;

Could Have:

- Different Search Engines – allow user an option to choose between search engines;
- Share Option – including emailing and other preferred social networks;
- Smart Results Appearing – preferable source appears first depending on star ranking, unpreferred sources appear randomly so that the user can interact with them;
- Meta-Tagging, **To be further discussed**;
- Sponsored Links – must filtered out from the search, no advertisement;
- Open Browser Function – allow an option to open search in a browser window, **To be further discussed**;

Won't Have:

- User Registration – no log in / sign in, user history is saved locally on device, **star ranking to be further discussed with IT client**;
- In-App Purchases

Some of the ideas have been deleted/drafted according to the client choice. The MoSCoW presented is valid for this term only as the next will require further discussions.

Requirements: Technical

'TermSeeker' User Scenarios

Basic Search Scenario

- The user opens the app.
- The user selects Language 1 (Language 1 is selected by default) and Language 2 (Language 2 is language chosen by default).
- The user enters some words in the search bar.
- The user also has certain keywords and attributes selected by default (or from the last search).
- The user types "word" in the search bar.
- The user makes basic search.
- The results will be shown on the screen sorted by their rankings.
- The user taps one of the links.
- When the user is done with the website, he/she taps on "Done" at top-right corner.
- The user has an option to rate the visited page from 1 to 5.
- The user can either choose a different link or perform another search.

OR

- The user opens the app.
- The user selects Language 1 only.
- The user has "Star Ranking" as sorting condition by default.
- The user also has certain keywords and attributes selected by default (or from the last search).
- The user types "word" in the search bar.
- The user makes basic search.
- The results will be shown on the screen sorted by their rankings.
- The user taps one of the links.
- When the user is done with the website, he/she taps on "Done" at top-right corner.

TermSeeker Use Cases

User Case: Basic Search

ID: 1

Brief Description: The user can do basic search.

Primary Actor: User

Secondary Actor: None

Preconditions: Internet Connection

Main Flow:

- The user selects one or two language(s) for the search.
- The user selects a sorting condition for the search.
- The user enters some words in the search bar.
- If the user chooses one language, the results in the specified language would be shown and would be sorted by the selected sorting condition.
- If the user chooses two languages, the results which have a content in both of the specified languages would be shown and would be sorted by the selected sorting condition.

Post Condition: None

Alternative Flow: If the user does not specify a language and/or a sorting condition, they will be set to English and by relevance respectively by default.

User Case: Advanced Search

ID: 2

Brief Description: The user can do advanced search by specifying some words.

Primary Actor: User

Secondary Actor: None

Preconditions: Internet Connection

Main Flow:

Technical Review Report

Report Structure: The report structure is as follows: major implementation as a heading, following small sections are steps numbered in order.

Search for words:

- Use Java or either Objective C or Swift languages for platforms Android and iOS.
- Implement Google mobile application.
- Add possibility to choose one or two words when having "English" set to default. This option allows greater flexibility for the user. e.g. choosing English will mean search through people on UK, choosing Russia - google.ru.
- Add table with the two words in .xml format (example of what). This will tell the key words to be searched and the order in which they appear.
- Add controls. This will mean to use "EditText", and such as through attributes in the search, scroll off or on by default. Later implementation of advanced mode will allow to tick off.
- Add possibility to choose second language. There are a few questions to be asked about this particular feature. Firstly, star-ranking requires database in order to have average score across all users. If there are two languages, then the average score for each language can be affected by our star-ranking system. Secondly, star ranking of our ranking system will mean that users may view the application in a different way for example, affecting the way they search for words. This will mean that the application will have to change the way it works. Finally, the application may end up costing companies expensive, though this is just my opinion. However, this is not a part of "Must Have" step for now, so let me know.
- Decide for iOS and Android - and test it. Might take some time, however, to eliminate some bugs and make search more stable.
- Add possibility to choose attributes. As a part of our work, we are to research attributes that

Advanced Search mode:

TermSeeker

- Write the bi-weekly report
- Final classes diagram
- Android
 - Implement Google search
 - Allow to choose keywords
 - Add Google attributes
 - Link languages to all data
- iOS
 - Joseph Xcode course
 - Joseph Xcode course

Item completed

Design Patterns: First Draft

Left Screen Mockup:

Language 1 Language 2

Search...

Search Results

Search User History Settings

Right Screen Mockup:

Language 1 Language 2

Search...

Key Words	Attributes
<input checked="" type="checkbox"/> Key Set 1	<input checked="" type="checkbox"/> All these words
<input checked="" type="checkbox"/> Key Set 2	<input checked="" type="checkbox"/> This exact word or phrase
<input checked="" type="checkbox"/> Key Set 3	<input checked="" type="checkbox"/> Any of these words
...	

Search User History Settings

Scrum

- an Agile framework for completing complex projects
- Two different applications
- Star Ranking system as a Database
- And so on...

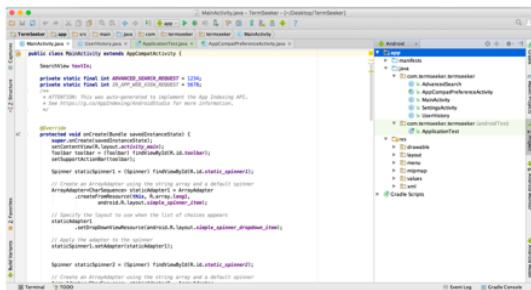


More Programming...

- We use two different language to compute: Swift (iOS) and Java (Android)
- Two different IDEs (Integrated Development Environments): Xcode and Android Studio
- Through requirements we can divide work and enter a project at almost any stage



Android Studio



A screenshot of the Android Studio IDE. The code editor displays Java code for an Activity named MainActivity. The code includes imports for Context, Intent, and Spinner, along with declarations for a Toolbar and a Spinner. The file structure on the right shows the project's directory structure under the 'Android' tab, including Java files like MainActivity.java and resources like drawables and layouts.

```

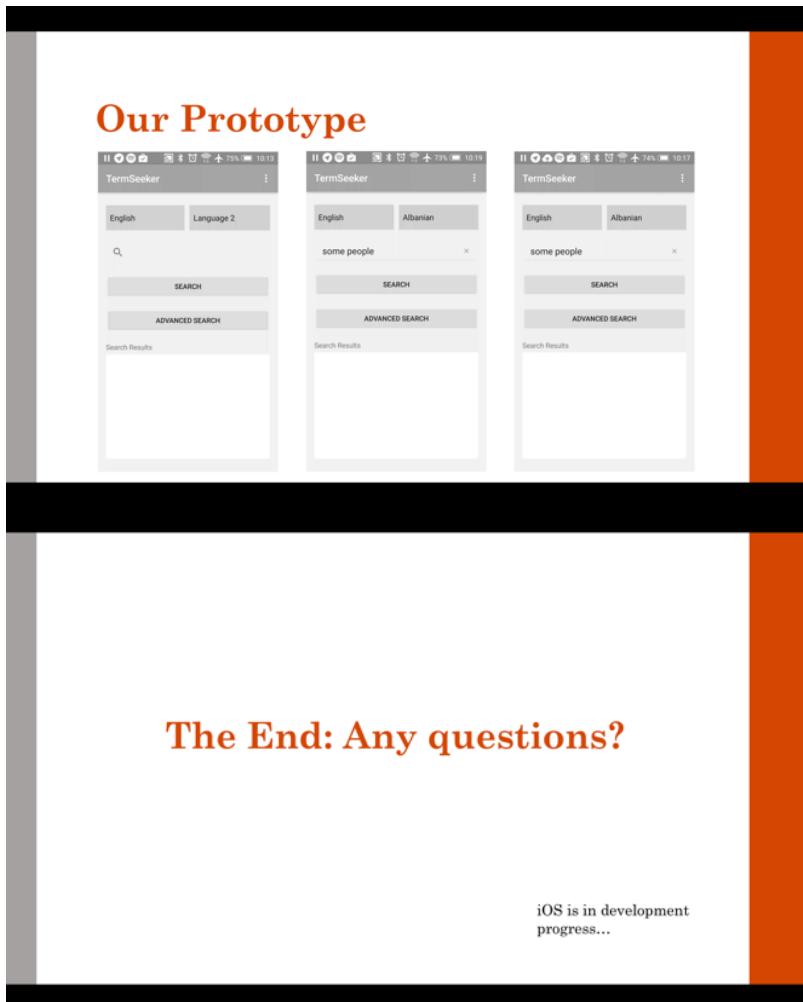
public class MainActivity extends AppCompatActivity {
    private Toolbar toolbar;
    private Spinner spinner;
    private static final int SPINNER_WIDTH_REQUEST = 1280;
    private static final int SPINNER_HEIGHT_REQUEST = 3670;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        spinner = (Spinner) findViewById(R.id.spinner);
    }

    @Override
    public void onConfigurationChanged(Configuration newConfig) {
        super.onConfigurationChanged(newConfig);
        spinner.setLayoutParams(new Spinner.LayoutParams());
    }
}

```



6.4 Logos

Here are the logos we draw for our client to choose from.



TermSeeker

6.5 Excel file

Column1	af	afrikaans	albanian	arabiese	armenian	azerbajani	eu	be	bn	bg	ca	zh-CN	hr	cs
ar	Afrikaans	افریکانس	البانيا	عربی	ارمنیه	azerbaijani	basque	Wit	বাস্ক	বালোসো	بلاروس	بلغاریه	كroatia	Tsjechie
az	afrikaans	albanian	arab	erməni	Azərbaycan	basque	Belarus	Bengali	Bolqar	catalan	çin	Xorvat	çek	
ba	afrikaans	albanian	arabisch	արմենի	ազերբայջանական	բասկա	belarusk	bulgar	bulgar	bulgarische	bulgarisch	bulgarische	bulgarisch	
bg	afrikaans	albanian	arabisch	արմենի	ազերբայջանական	բասկ	belarusk	bulgar	bulgari	bulgarisch	bulgarisch	bulgarisch	bulgarisch	
bn	আফ্রিকান্স	আলবানিয়ান	আরবি	արմեն	আজেরবাইজান	বাশক	বেլারুস	বার্জিন	বার্জিন	বাস্ক	বাস্ক	বাস্ক	বাস্ক	
ca	afrikaans	albanian	árab	armeni	azerí	basc	bielorús	bengali	búlgar	bulgàr	bulgàr	bulgàr	bulgàr	
cs	afrikáns	albánsky	arabský	arménský	azérbájdžánský	basque	béloruské	bengalský	bulharský	bulharský	bulharský	bulharský	bulharský	
cy	Africans	Albaneg	Arabeg	Armeneg	azərbaycanlı	Basqeg	Belaruseg	Bengalieg	Bulgareg	Bulgareg	Bulgareg	Bulgareg	Bulgareg	
da	afrikaans	albanian	arabisch	armensk	aserbajdsjansk	baskisk	hviderussisk	bengali	bulgarsk	bulgarsk	bulgarsk	bulgarsk	bulgarsk	
de	afrikaans	albanian	arabisch	armenisch	azərbaycanlı	Baschisk	burjatisk	Bulgarsk	Bulgarsk	Bulgarsk	Bulgarsk	Bulgarsk	Bulgarsk	
el	Αφρικανού	Αλβανικό	Αραβικό	Αρμενικό	Ազերբայջանական	Βόρδικα	Λευκορωσικά	Μπενγκάλ	Βούλγαρικό	Κοτούνικού	Κυνέζικ	Κροατικό	Τσχεχικό	
en	afrikaans	Albanian	Arabic	Armenian	Azerbaijani	Basque	Belarusian	Bengali	Bulgarian	Catalan	čína	Croatian	Czech	
eo	afrikaans	Albana	Araba	Armenian	Azerbaijan	eúskua	Belarusian	Bengali	Bulgara	Cataluna	čína	Kroata	čeha	
es	afrikaans	albanian	árabe	armenio	azerí	vuscene	belorruzo	bengali	búlgaro	catalán	china	croata	checo	
et	afrikaani	albaania	arabis	armeenia	aserbaidzaani	baski	Galvegene	bengali	bulgaria	Katalaani	hiina	hovraadi	šehe	
eu	afrikaans	albanian	arablera	armenian	azerbajani	eukraian	belarusian	Bengali	bulgarian	catalan	txikako	Kroaziako	Txekiar	
fa	افریکانس	البانيا	عربی	ارمنی	آذربایجان	بلاروس	بنگالی	بلاری	بلاروس	کاتالان	چینی	کرواتی	چک	
fi	afrikaans	albanialainen	arabialainen	armenian	Azerbaidzhanin	baski	Valko	Bengali	bulgariainen	katalaani	kiinalainen	koontsatti	Tskekin	
fr	afrikaans	albanais	arabe	arménien	azerbaïdjanais	basque	blélorusse	bengali	bulgare	catalan	chinois	croate	tchéque	
ga	afrikaans	Albáníos	árabe	arménios	Asásbóláis	Bascais	Bealarús	Beangális	Bulgáris	Catalánis	Sínis	Cróitís	Seice	
gl	afrikaans	albanés	árabe	armenio	azeraibaidxanés	bieloruso	bengali	bulgáro	bulgáro	catalán	chines	croata	checo	
hi	আফ্ৰিকান্স	আলবানিয়ান	আরবী	অৰ্মেনীয়	আজেৰবাইজান	বেলোৱু	বেলোৱু	বেলোৱু	বুলগুৰ	কাতালান	চিন	ক্রোত	চেক	
hr	afrikaans	Albanac	Arapski	Armenac	Azerbaijanac	Basjkirski	Bełoruski	Bengalski	Bulgarski	Catalan	čínska	hrvatski	češki	
hu	afrikaans	albanian	Arabic	armenian	azerbaijani	basque	belarusian	Bengali	bulgarian	Catalan	Chinwa	croatan	czech	
hy	աֆրիկանս	ալբանաց	արաբաց	արմենաց	ազերբայջանաց	բաշক	փেհেրօր	bengali	bulgár	catálán	kinai	horvát	cseh	
id	Afrikansas	Albanian	Arabisch	Armenisch	Azerbaijani	Baskisch	burjatisch	Bengali	Bulgarija	Catalan	Qhínsilvánip	Hungríash	čítan	
is	afrikaans	albanian	Arabic	armenian	azerbajiani	basque	burjatian	Bengali	Bulgarian	Catalan	Cina	Kroasia	Ceko	
it	afrikaans	albanese	arabo	armeno	azero	basco	burjatian	Bengali	Bulgarego	Catalano	Chinese	croatan	czescho	
iw	אַפְרִיקָאנְס	אַלְבָנִיאָנִי	אַרְבָּיִתְ	אַרְמָנִיְתְ	אַזְּרְבָּיְזָנִיְתְ	בָּשָׂקֵעַ	בְּלָרְוָסָה	בְּלָרְגָּלְגָּלְסָה	כָּטָּלָןְ	כִּינְסָה	כְּרֹוְתָּאָצָה	כְּצֵחָה		
ja	アフリカーン	アルバニア語	アラビア語	アルメニア語	アゼルバイジャン語	ベラルーシ語	ベンガル語	ペルギラシ語	ペンガル語	ペルギラシ	中国語	クロアチア語	チエニコ語	
ka	afrikaans	albanian	arabische	armenische	Azerbaidschanische	basque	Balkanische	Bengali	bulgariainen	catalaani	kiinalainen	koontsatti	Tskekin	
ko	아프리카	알바니아	아라비아	알마니아	아제르바이잔	바스크	발가리아 사람	발가리아 사람	발가리아	발가리아 사람	발가리아 사람	발가리아 사람	발가리아 사람	
la	Africa	Albanian	Arabic	Armenian	Azerbaijan	Basque	Belarusian	Bengali	bulgare	Catalan	chinois	croate	tchéque	
lo	afrikaans	albanian	arabische	armenische	azeraibaidzhanische	bieloruso	Bengali	bulgari	bulgari	Catalan	chiros	croatis	czec	
lt	afrikansas	albanian	arabu	armenė	azeraibaidzhanie	Basqu	Baltarusijos	Bengali	bulgari	Catalan	kinios	croat	czekios	
lv	afrikaū	albānū	arabu	armēni	azeraibaidzhanis	basku	Balkrievijas	Bengali	bulgarijs	Catalan	kiniešu	croat	čehu	
mk	afrikans	albanicki	arapski	armenski	azeraibajdzanski	baski	beloruski	Bengali	bulgarijski	Catalan	kiniešu	croat	čehsko	
ms	Afrikaans	Bahasa Arab	Armenia	Azerbaijan	Azerbaijan	Basque	Belarusian	Bengali	bulgarijski	Catalan	Cina	Croatian	Czech	
mt	afrikaans	Albanit	Għarbi	Aħnejha	Ażżejt	Bask	Belarusian	Bengali	bulgarijski	Catalan	Ciniz	Croat	Ceka	
nl	afrikaans	Albanian	Arabisch	Armeens	Azerbaijanse	Basqu	Baltarusijos	Bengali	bulgarijski	Catalan	Cinjist	Croatian	Czech	
no	afrikaans	albanian	arabic	armenian	azerbajdzhan	basque	belarusian	Bengali	bulgarijski	Catalan	kinios	croat	czekios	
pl	afrikaans	albánski	arabski	armenski	azeraibajdzanski	basquski	belaruski	Bengali	bulgarijski	Catalan	kinios	croat	čekos	
pt	afrikaans	albanian	árabe	arménio	azerbaidjanes	basco	belarusiano	Bengali	bulgarijski	Catalan	kinios	croat	česko	
ro	afrikaans	albanez	arabis	armean	azero	basco	belarusian	Bengali	bulgarijski	Catalan	chinez	croat	čeh	
ru	африканськ	албанськ	арабськ	арменськ	азербайджанськ	баскськ	беларуськ	бенгальск	булгарськ	каталан	кітайськ	хорватськ	чеськ	
si	afrikaans	albanian	arabische	armenische	azeraibajdzanski	basque	belarusian	Bengali	bulgarijski	Catalan	китай	хорват	чески	
sk	afrikáns	albánsky	arabský	arménsky	Azerbájdžanský	basque	beloruský	Bengali	bulgarijský	Catalan	кітай	хорват	ческ	
sl	afrikaans	albánska	arabska	arménska	Azerbájdžanská	basvka	belorusčina	Bengali	bulgarijska	Catalan	кітай	хорват	ческа	
sq	afrikaans	shqiptar	arab	armeni	azeraibajdzan	basv	beloruski	Bengali	bulgarijski	Catalan	кітай	хорват	ческа	
sr	afrikaans	albánski	arabski	armenski	azeraibajdzanski	basvski	beloruski	Bengali	bulgarijski	Catalan	кітай	хорват	ческа	
sv	afrikaans	albansk	arabisk	armensk	azeraibajdzansk	basvsk	beloruski	Bengali	bulgarijski	Catalan	кітай	хорват	ческа	
sw	afrikaans	Kiellänen	Kiarabu	Kiarmeni	Kiärmänen	Kiärmänsk	Kiärmänsk	Kiärmänsk	Kiärmänsk	Katalan	Kiärmänsk	Kiärmänsk	Kiärmänsk	
ta	ஆஃப்ரிக்கன்	ஆல்஬ானியன்	ஆர்வை	ஆர்மீனி	ஆজெர்஬ைஜானியன்	ஆஸ்கா	ஆபார்டு	ଆପାର୍ଟିମେଣ୍ଟ	ଆପାର୍ଟିମେଣ୍ଟ	ଆପାର୍ଟିମେଣ୍ଟ	ଆପାର୍ଟିମେଣ୍ଟ	ଆପାର୍ଟିମେଣ୍ଟ	ଆପାର୍ଟିମେଣ୍ଟ	
te	ఆఫ్రికాన్	అలబానియన్	అర్బి	అర్మెని	అజెర్బాయజాన్	బాస్కు	అబ్రాస్	అబ్రాస్	అబ్రాస్	అబ్రాస్	అబ్రాస్	అబ్రాస్	అబ్రాస్	
th	afrikaans	albanian	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	ພ່າຍ	
ti	afrikaans	Albanes	Arabe	Armeny	Azerbaijan	Basko	Belarusian	Bengali	Bulgarian	Catalan	Tsino	Kroato	Tsiek	
tr	Afrika Dilleri	Arnavutça	Arapça	Erməniece	Azərbaycan	Filippino	Filipino	Filipino	Filipino	Filipino	Filipino	Filipino	Filipino	
uk	африканськ	албанська	арабська	арменська	азербайджанська	баскська	белоруська	белоруська	белоруська	белоруська	белоруська	белоруська	белоруська	
ur	اپریکان	البانی	عربی	ارمنی	اوزبک	بازکس	بلاروسی	بلاروسی	بلاروسی	بلاروسی	بلاروسی	بلاروسی	بلاروسی	
vi	Afrikaans	Albania	Tiếng Rô	Armenia	Azerbaijan	Basque	Bela	Bengali	Bulgaria	Catalan	Tiếng Trung	Croatia	Séc	
yi	אַפְרִיקָאנְס	אַלְבָנִיאָנִי	אַרְבָּיִתְ	אַרְמָנִיְתְ	אַזְּרְבָּיְזָנִיְתְ	בָּשָׂקֵעַ	בְּלָרְוָסָה	בְּלָרְגָּלְגָּלְסָה	כָּטָּלָןְ	כִּינְסָה	כְּרֹוְתָּאָצָה	כְּצֵחָה		
zh-CN	南非荷蘭語	阿尔巴尼亞语	阿拉伯语	亚美尼亚语	阿塞拜疆语	阿塞拜疆语	白俄罗斯语	孟加拉语	孟加拉语	佩加利耶语	加泰罗尼亚语	中文	克罗地亚语	
zh-TW	南非荷蘭語	阿爾巴尼亞語	阿拉伯語	亞美尼亞語	阿塞拜疆語	阿塞拜疆語	白俄羅斯語	孟加拉語	孟加拉語	佩加利耶語	加泰羅尼亞語	中文	捷克语	
zz	الغافر	اللبنانية	العربية	الآرمنية	الأذربيجانية	الباسكية	البلاروسية	البلغارية	البلغارية	الكتالانية	الصينية	الكرواتية	التشيكية	

english	dictionary	glossary	lexicon	vocabulary	terminology
af	woordeboek	woordelys	leksikon	woordeskat	terminologie
ar	قاموس	معجم	معجم	مفردات	مصطلحات
az	lüğət	glossary	lexicon	lügət	terminologiya
be	слоўнік	глазарый	лексіка	лексіка	тэрміналогія
bg	речник	речник	лексика	лексика	терминология
bn	অভিধান	শব্দকোষ	শব্দকোষ	শব্দভান্ডার	পরভিয়া
ca	diccionari	glossari	lèxic	vocabulari	terminologia
cs	slovník	glosář	slovník	slowník	terminologie
cy	geiriadur	geirfa	geirfa	geirfa	termau
da	ordbog	ordliste	leksikon	ordforråd	terminologi
de	Лෝජ්සිකොම්බ ත්‍රේම්ඩිජ්බඩාස		lexicon	ලෝජ්සිකා	ත්‍රේම්ඩිංජ්ඩාස
el	λεξικό	γλωσσάρι	λεξικό	το λεξιλόγιο	την ορολογία
en	dictionary	glossary	lexicon	vocabulary	terminology
eo	vortaro	glosaro	leksikon	vortprovizo	terminologio
es	diccionario	glosario	léxico	vocabulario	terminología
et	sõnastik	sõnastik	leksikon	sõnavara	terminite
eu	hiztegia	glosario bat	hiztegia	hiztegi	terminologia
fa	فرهنگ لغت	واژه نامه	واژه نامه	فرهنگ لغت	اصطلاحات
fi	sanakirja	sanasto	sanasto	sanasto	terminologiaa
fr	dictionnaire	glossaire	lexique	vocabulaire	terminologie
ga	Foclóir	gluais	lexicon	stór focal	téarmaíocht
gl	dicionario	glosario	léxico	vocabulario	terminoxía
hi	শব্দকোষ	শব্দাবলী	শব্দকোশ	শব্দাবলী	শব্দাবলী
hr	rječnik	pojmovnik	leksikon	rječnik	terminologije
ht	diksyonè	glosè	leksik	vokabiliè	tèminoloji
hu	szótár	szószedet	lexikon	szótár	terminológia
hy	բառարան	բառարան	բառապաշտ	բառապաշտ	տերմինարանության
id	kamus	glosarium	leksikon	kosakata	istilah
is	orðabók	ordasafn	Lexicon	orðaforða	hugtök
it	dizionario	glossario	lessico	vocabolario	terminologia
iw	מילון	מילון מונחים	לקסיקון	אוצר מילים	טरמינולוגיה
ja	辞書	用語集	語彙	語彙	用語
ka	Wörterbuch	Glossar	Lexikon	Wortschatz	Terminologie
ko	사전	용어	사전	어휘	용어
la	dictionary	glossario	Lexicon	vocabulorum	terminologia
lo	ຈາກນິວໜີນ	ສັບ	ຄ້າສັບ ຄ້າສັບທາງນີ້ ທະນາຍົດນີ້ ໄລຍະຍົງ	ຄ້າສັບ	
lt	žodynėlis	žodynėlis	žodynės	žodynės	terminologija
lv	vārdnīca	glosārijs	leksika	vārdu krājums	terminoloģija
mk	речникот	речник	лексикон	вокабулар	терминологија
ms	kamus	glosari	kosa kata	perbendaharaan kata	istilah
mt	dizjunarju	glossarju	lessiku	vokabolarju	terminologija
nl	woordenboek	verklarende woordenlijst	lexicon	woordenschat	terminologie
no	ordbok	ordliste	leksikon	vokabular	terminologi
pl	słownik	słownik	słownictwo	słownictwo	terminologia
pt	dicionário	glossário	léxico	vocabulário	terminologia
ro	Dicționar	glosar	lexicon	vocabular	terminologie
ru	словарь	глоссарий	лексика	лексика	терминология
sk	slovník	glosár	slovník	slovník	terminología
sl	slavar	glosar	leksikon	besednjak	terminologija
sq	fjalor	fjalorth	leksikun	fjalori	terminologjinë
sr	речник	речник	лексикон	лексика	терминологија
sv	ordbok	ordlista	lexikon	ordförråd	terminologi
sw	kamusi	glossary	Lexicon	msamati	maneno
ta	அகராதி	அருஞ்சொற்பொருள் இத்தெ		சொல்லகராதி	சொல்ல
te	నిఘటువు	పదక్షిం	క్షిం	పదహాలం	పశ్చాత్
th	ພຈນາບຸກຮມ	គຳສົ່ວໂພ	ສ່ວ	ສ່ວ່ພ	ສ່ວ່ພແສງ
tl	diksyonaryo	glossary	leksikon	bokabularyo	terminolohiya
tr	sözlük	sözlük	sözlük	kelime	terminoloji
uk	словник	глосарій	лексика	лексика	термінологія
ur	ڈکشنری	ابوان کی لغت	ڈکشنری	ذخیرہ الفاظ	اصطلاحات
vi	từ điển	chú giải	từ vựng	từ vựng	thuật ngữ
yi	גָּוָרְבָּוּרְטָרְפָּאָרְ	וּעָרְבָּוּרְטָרְפָּאָרְ	לְקָסְטִיקָאָן	אַקָּאָבָּוּלָּאָרְ	טַעֲרַמִּינְאָלָּאָגָּעָ
zh-CN	术语词典	词库	词汇	短期银行	术语库

thesaurus	jargon	word list	term base	term bank	word bank	بنك الكلمة	LANGUAGE NAME (EN)	CODE
tesourus	jargon	woordelys	termyn basis	termyn bank	woord bank	بنك المدى	Afrikaans	af
thesaurus	jargon	söz siyahisi	müddatlı bazası	müddatlı bank	söz bank	قاعدة المدى	Albanian	sq
тезаурус	жаргон	список слов	термин база	терминът банка	слова банк	词汇表	Arabic	ar
синонимен речник	жаргон	списък от думи	терминологична база	терминът банка	думата банка	词汇表	Armenian	hy
জ্ঞানভূক্তির	অভিধান	শব্দ ভালকা	শব্দটো বাস	ঘণ্টার শব্দ	শব্দ বাস	词汇表	Azerbaijani	az
diccionari de sinònims	argot	llista de paraules	base de terme	banc termini	banc de paraules	词汇表	Basque	eu
tezaurus	žargon	seznam slov	termín základna	termín banka	Světové banky	词汇表	Belarusian	be
thesawrws	jargon	rhestr geiriau	sylfaen tymor	banc tymor	banc geiriau	词汇表	Bengali	bn
synonymordbog	jargon	ordliste	termbase	term bank	ord bank	词汇表	Bulgarian	bg
თექსურული	კარგობი	ხელყვა ხას	ტერმინი ბაზა	ტერმინი ბაზი	სიტყვა ბაზი	词汇表	Catalan	ca
του θησαυρού	ορολογία	λίστα έξειων	βάση όρων lexikó	βραχιοπόθεσμες τραπεζικές τράπεζα λέξη	τράπεζα λέξη	词汇表	Chinese	zh-CN
thesaurus	jargon	word list	term base	term bank	word bank	词汇表	Croatian	hr
tezáuro	jargon	vortlisto	termino bazo	termino banko	vorto banko	词汇表	Czech	cs
diccionario de sinónimos	jerga	lista de palabras	base de términos	banco de términos	banco de palabras	词汇表	Danish	da
thesaurus	kõnepruugis	sõnade	terminibaasina	Pangalaenude	sõna park	词汇表	Dutch	nl
thesaurus	jargon	hitz zerrenda	epe base	epe banku	hitza banku	词汇表	English	en
اصطلاحات		الكلمة	ليس كذلك	پایه مدت	بانک مدت	词汇表	Esperanto	eo
sanasto	slangia	sana luettelo	termi pohja	termi pankki	sana pankki	词汇表	Estonian	et
théaurus	le jargon	liste de mots	base de termes	bancaire à terme	banque de mots	词汇表	Filipino	tl
teásárlis	béarlagair	liosta focal	bonn téarma	téarma bainc	banc focal	词汇表	Finnish	fi
enciclopedia	argot	lista de palabras	base prazo	banco prazo	base de palabras	词汇表	French	fr
কোর্স	শব্দজ্ঞাল	শব্দ সমূহ	অবধি জাপার	অবধি কেক	শব্দ কেক	词汇表	Galician	gl
tezaurus	žargon	popis riječi	pojam baze	izraz banka	rijec je banka	词汇表	German	de
sinonim	jagon	lis mo	tém baz	tém labank	bank mo	词汇表	Georgian	ka
szinonimaszótár	zsargon	szó listát	terminológiai adatbázisa	lejrátúr banki	szószedet	词汇表	Greek	el
թեզունուն	դարպոն	բազող ցուցակը	ճամկետ բազման	ճամկետ բազման	Բազ բակալյան	词汇表	Haitian Creole	ht
thesaurus	jargon	dafta kata	basis jangka	bank jangka	Bank kata	词汇表	Hebrew	iw
samehtaorðabók	hrognamál	orðalista	grunnur	tíma banka	orðið banki	词汇表	Hindi	hi
thesaurus	gergo	elenco di parole	termijn basis	banka termini	lista di termini	词汇表	Hungarian	hu
תֵּאַרְוָן	גָּרוֹגָן	עִירָּוֶת מִילְוָה	בָּסְטוּוֹן	בָּקָטָה	מִילָּת בָּנָךְ	词汇表	Icelandic	is
シソーラス	専門用語	単語リスト	用語ベース	用語の銀行	単語銀行	词汇表	Indonesian	id
Thesaurus	Jargon	Wortliste	Terminologiedatenbank	tigen Bank	Wort Bank	词汇表	Irish	ga
동의어 사전	전문 용어	단어 목록	용어의 기초	용어 은행	단어 은행	词汇表	Italian	it
thesaurus	turpiloquio	verbum codex	term base	term ripam	verbum ripam	词汇表	Japanese	ja
thesaurus	չունչառական	նամակաշախաղ ջայուղական	միացու լուսական	շահագու ժամանակ	շահագու ժամանակ	词汇表	Korean	ko
tezauras	žargonas	žodis sraške	terminų bazė	terminų bankas	Pasaulio banko	词汇表	Lao	lo
težauru	žargons	várdú sarakstu	terminū bázē	termins banka	várdús banka	词汇表	Latin	la
речник	жаргон	листа на зборови	терминот базен	терминот банка	зборот банка	词汇表	Latvian	lv
thesaurus	jargon	senarai perkataan	asas jangka	bank jangka	bank perkataan	词汇表	Lithuanian	lt
težawru	jargon	lista kelma	baži terminu	bank terminu	bank kelma	词汇表	Macedonian	mk
thesaurus	jargon	woordenlijst	termijn basis	woord bank	woord bank	词汇表	Malay	ms
synonymordbok	sjargong	ordliste	sikt base	term bank	ordet bank	词汇表	Maltese	mt
tezaurus	žargon	lista słów	baza termin	banku termin	Bank slowo	词汇表	Norwegian	no
encyclopédia	jargão	lista de palavras	base prazo	banco prazo	banco de palavras	词汇表	Persian	fa
tezaur	jargon	lista de cunverte	de bază pe termen	bancare pe termen	banca cuvânt	词汇表	Polish	pl
тезаурус	жаргон	список слов	термин база	срок банк	слово банк	词汇表	Portuguese	pt
thesaurus	žargon	zoznam slov	termín základňa	termín banka	Svetovej banky	词汇表	Romanian	ro
besednjak	žargon	seznam besed	izraz baza	izraz banka	beseda banka	词汇表	Russian	ru
enciklopedi	zhargon	lista fjalë	baza afat	banka termi	banka fjalë	词汇表	Serbian	sr
тезаурус	жаргон	Ворд Лист	појам база	термин банка	реч банка	词汇表	Slovak	sk
synonymordbok	jargong	ordlista	termen bas	termbank	ord bank	词汇表	Slovenian	sl
Thesaurus	jargon	orodha neno	neno msingi	mrefu benki	neno benki	词汇表	Spanish	es
କଣ୍ଠାର୍ଥିଚିପତ୍ରଣୀ	ବାକ୍ସକନ୍କର୍ଣ୍ଣ	ବାର୍ଗର୍ତ୍ତତମ ପଟ୍ଟିମାତ୍ର	କାଳ ଅପିପଟଣ	କାଳ ବାନ୍ଦି	ବାର୍ଗର୍ତ୍ତତମ ବାନ୍ଦିଯିଲ୍	词汇表	Swahili	sw
ଦିନାର୍ତ୍ତ	ଅବେଳ୍ୟମ୍	ପଦାଳ୍ୟମ୍	ପଦା ଲୀନ୍	ଫାର୍କ୍ ଲୀନ୍ୟାଙ୍କ	ପଦା ଲୀନ୍ୟାଙ୍କ	词汇表	Swedish	sv
ରାଜାର୍ଦା	ରେଖ	ପରିବାରମ	ରାଯାରାର୍ଦା	ରାଯାରାର୍ଦା	ରାଯାରାର୍ଦା	词汇表	Tamil	ta
thesaurus	magulong pag-uusap	listahan ng salita	term base	term bank	bangko ng salita	词汇表	Telugu	te
hazine	jargon	kelime listesi	termin kökeni	term bankası	kelime bankası	词汇表	Thai	th
тезаурус	жаргон	список слів	термін база	термін банк	слово банк	词汇表	Turkish	tr
قاموس	شيدحال	لطف کی فہرست	مدت کی بنیاد	مدت کی بنیاد	اطلاع بنک	لغط بنک	Ukrainian	uk
từ điển đồng nghĩa	thuật ngữ	danh từ	cơ sở hạn	hạn ngân hàng	ngân hàng từ	لغط بنک	Urdu	ur
စାହାର୍ଥୁ	ଅଗାରା	ଶାମା ଟାରା	ଉତ୍ତମ	ଉତ୍ତମ	ଉତ୍ତମ	لغط بنک	Vietnamese	vi
彳话	单词列表	术语	词库	词库	词库	لغط بنک	Welsh	cy

The report is fully written and edited by Daniil Gannota.